DATA FOR THE COMMON GOOD

GSOC 2024 Proposal

D4CG: Front-end and Adaptive Question Rendering (Project 1)

KOLANU VARUN

D4CG: Front-end and Adaptive Question Rendering

GSOC 2024

Name: Kolanu Varun
Major: Computer Science and Engineering
Degree: Bachelor of Technology
Year: Sophomore
University: Indian Institute of Technology, BHU (Varanasi)
GitHub Handle: https://github.com/Varun-Kolanu

Postal Address: 20-120/A, Street No.4, Sharada Nagar, Saroornagar, Hyderabad, Telangana (500035), India.
Email: kolanuvarun739@gmail.com
Phone: (+91) 9603138312
Resume: Link
Timezone: Indian Standard Time (UTC +5:30)

About Me:

Hi! Myself Kolanu Varun, a Computer Science Engineering sophomore from IIT (BHU) Varanasi, India. I like to deep dive into various technologies, proficient primarily in Web Development. I have experience in full stack development and have contributed to organizations Zulip and D4CG. I have led teams in various Institute fests and also participated in hackathons. Grabbed Bronze medal in Inter IIT Tech Team 12.0 in Web Development domain.

My Contribution:

I have made a sample project which shows my implementation. Link to the repo: Link

Proposal Abstract:

.

The aim of this project is to enhance the user-friendliness of D4CG clinical trial matching

application, particularly targeting individuals without medical expertise. This will be achieved by incorporating an algorithm to prioritize the questions presented to the user. The project will focus on optimizing and implementing the REACTJS web application component, enabling the rendering of various question types with different input formats.

An adaptive web form capable of dynamically adjusting based on a JSON input will be developed, and this new component will be integrated into the main application.
There is much scope of improvement in the UI of the clinical trial matching application to make it user-friendly.

Proposal:

1. Architecture Design:

React JS will be used as frontend framework. ReactJS naturally lends itself to a component-based architecture, which promotes modularity and reusability. Each React component can encapsulate its own logic and UI, making it easier to understand, maintain, and scale.
React components enable a clear separation of concerns by dividing the UI into small, reusable pieces. State Management in React is efficient. Redux can also be used.
Hence React JS will be an optimal option for creating the required component as it can be easily integrated into the existing application.

2. Data Structure Design:

I. Questions:

JSON schema can be used to represent the Questions, their labels and values:

Present implementation in the existing application:

```
{
    "id": 1,
    "groupId": 1,
    "name": "age",
    "min": 0,
    "label": "What is the patient's current age (in years)?",
    "type": "number"
```

The JSON structure of the questions can be changed according to the requirements which can be discussed.

II. Clinical Trials:

The clinical trial eligibility criteria are in the form of an AND – OR tree with the Questions as nodes and the values as branches. This tree structure can be represented in JSON format cleverly.

General information like title, description, location can also be represented in JSON.

To represent the eligibility criteria:
Let's take an example from the existing application of the clinical trial **2020-0484:**

```
( What is the patient's current age (in years)? is less than/equal to 21 ) AND
(
        (
                ( What is the patient's current diagnosis? is equal to "Acute myeloid leukemia (AML)" ) AND
                (
                        (
                                ( Is the patient currently in relapse (or suspected relapse)? is equal to "Yes" ) AND
                                ( Does the patient currently have, or have they in the past had, confirmed or suspected relapse disease? is
                                equal to "Yes" ) AND
                                ( How many confirmed or suspected relapses, including the current if applicable? is greater than/equal to 1 )
                        ) OR
                        (
                                ( Is the patient's disease currently refractory? is equal to "Yes" ) AND
                                ( Does the patient currently have, or have they in the past had, refractory disease? is equal to "Yes" ) AND
                                ( How many occurrences of refractory disease, including the current if applicable? is greater than/equal to 1 )
                        )
                )
        ) OR
        (
                What is the patient's current diagnosis? is equal to "Treatment-related acute myeloid leukemia (tAML)" AND
                Does the patient currently have, or have they in the past had, confirmed or suspected relapse disease? is equal to "No" AND
                Is the patient currently in relapse (or suspected relapse)? is equal to "No"
        )
) AND ...
```

This can be converted into JSON as:


```
"trials": [
    {
        "criteria": [
            {
                "operator": "and",
                "conditions": [
                    {
                        "age": {
                            "max": 21
                        }
                    },
                    {
                        "operator": "or",
                        "conditions": [
                            {
                                "operator": "and",
                                "conditions": [
                                    {
                                        "diagnosis": "Acute myeloid leukemia (AML)"
                                    },
                                    {
                                        "operator": "or",
                                        "conditions": [
                                            {
                                                "operator": "and",
                                                "conditions": [
                                                    {
                                                        "ever_refractory": "Yes"
```

1. Each trial will be an array of one JSON either of operator “and” or “or”.
1. Each **JSON** represents a **Boolean** value / condition that will be evaluated based on the user’s input to questions.

2. There will be **three** types of JSONs. One is with the operator “**and**”, second with operator “**or**”, third with no operator (a **single** Boolean value).

3. Single JSONs can be of many types. For example, if it is of **numeric** type, **max** and **min** of the input can be defined. The **key** of this JSON will be the “**name**” we defined in the Questions JSON schema, and the **value** of it represents the **constraint** of the user’s input on which the Boolean will be evaluated to True.

   If the Question’s value is of **Select** or **Radio** button type, the value can be represented as a “**String**”. This string represents that the Boolean gets evaluated to true only when the input of this Question is equal to the value defined in the JSON (which is the label defined in the Questions JSON.

4. The JSON with operators “and”, “or” will have conditions in an array. This array represents the operation “and”, “or” applied on all the Booleans inside the array respectively.

   For example, the Boolean (x & y & z) can be represented as
   {
   “operator”: “and”,
   “conditions”: [x, y, z]
   }

   Where x, y, z are JSONs like {“diagnosis”: “Acute myeloid leukemia (AML)”}
   Which evaluates to true if the user’ input of diagnosis will be Acute myeloid leukemia (AML).

3. Algorithm Implementation:

The present implementation of asking user the questions needed for showing the matched trials is very static. The implementation asks all the questions at a time but it should be dynamic, according to the input from the user.

My proposed Algorithm:

I. Choosing the Questions:

Questions to be asked can be chosen according to the Hierarchical pattern of the tree and based on properties of Boolean Logic. This works on the fact that “Ask only those questions which determine their chance of matching or not matching clinical trials”.

While traversing the tree from root to leaf (i.e., traversing the criteria in the JSON), we shall choose questions.

i. If we stumble upon the “and” operator JSON, traverse the conditions array **recursively** until we reach a single JSON (without any operator). Then add this question to the question list, come out of this “and” JSON and continue the traversal.
Reason: The “and” operator returns false if any of the Boolean is false. So, the number of questions to be asked reduces.

```
export function findEssentialQuestions(trials) {
    let essentialQuestions = [];

    function traverseCriteria(criterion) {
        if (criterion.operator === "and") {
            // Look for single conditions in "and"
            for (const condition of criterion.conditions) {
                if (typeof condition !== "boolean") {
                    if (!condition.operator) {
                        // Single condition found, add its key to essential questions
                        const key = Object.keys(condition)[0];
                        essentialQuestions.push(key);
                    }
                    else {
                        traverseCriteria(condition)
                    }
                    break;
                }
            }
```

```
    } else if (criterion.operator === "or") {
        // Look for single conditions in "or"
        for (const condition of criterion.conditions) {
            if (typeof condition !== "boolean") {
                if (!condition.operator) {
                    // Single condition found, add its key to essential questions
                    const key = Object.keys(condition)[0];
                    essentialQuestions.push(key);
                }
                else {
                    traverseCriteria(condition)
                }
            }
        }
    }

}

trials.forEach(trial => {
    traverseCriteria(trial.criteria[0])
})

essentialQuestions = Array.from(new Set(essentialQuestions))
return essentialQuestions;
```

II. Handling User Input:

When the user changes input, we shall re-evaluate Booleans in the trials.

i. We'll re-evaluate only those question names' Booleans, which are currently asked.
ii. If any single JSON is matched with user input, that JSON will be replaced by true. If not matched, will be replaced by false.
iii. If any of the Boolean in the "and" array evaluates to false, the whole "and" JSON is replaced by false
iv. If any of the Boolean in the "or" array evaluates to true, the

whole “or” JSON is replaced by true.

v. After the re-evaluation, questions are again selected according to the Step 1 (Choosing questions).

III. Showing Matched Trials:

i. There will be a key “matched” in trial’s JSON.

ii. If the type of trial.criteria[0] is **true** the trial is **matched**, if **false** it is **Unmatched**, and if neither (i.e., **object** type), trial is Undetermined.
Reason: If every evaluation of a trial is completed, the criterion array will have only one Boolean representing the status of match. If the traversal is not over yet fully, it will be still an object and hence we can’t determine the match status now.

IV. End of Questions:

The questions will be stopped when the count of Undetermined questions will be zero. Hence by this approach, number of questions that should be asked reduces significantly.

- We can think upon a different and efficient algorithm which is not only user friendly but also efficient in traversal, computation for large dataset.
- AI model can be integrated in this which takes the inputs of user and returns questions related to the previous responses.

4. Component Implementation:

I. The selected questions are rendered with the help of selected question names, and also the Questions available to us.

II. User selects the input of questions and the questions are rendered based on the users' input. But the questions which are selected already are preserved their position on the screen so that users can edit them whenever necessary.

III. The matched, undetermined, unmatched questions, their count are shown dynamically on the screen.

IV. Clean code practices and folder structure are used.

V. Necessary comments and good documentation are written parallelly.

VI. The component is integrated into the main application at last.

VII. **Accessibility** features like for **screen readers** will be implemented

5. <u>Questions Modification:</u>

1. New Questions can be chosen which are written in simpler language so that anyone can understand them easily

2. AI models can be integrated into the website which takes the input from the user and generate new questions on the go according to relevance of medical condition of the user, clinical trials and which generates a simpler text.

3. Users can ask AI (or any LLM model we use) to explain a certain question for simpler understanding.

6. Visualizations for complex data:

I. Graphs and pie charts can be used to show the users complex trial matching decisions.

II. The eligibility criteria shown will be picturized in a beautiful way like a tree or any decided UI.

7. Progressive Web App (PWA) support and caching:

1. Introduce PWA support so that users can use it offline too.

2. Use caching and memoization techniques for efficient data fetching and traversal of data.

Proposed Timeline

1. **Community Bonding Period:**

   - Will research more about the best practices in ReactJS, clean code practices.
   - Will reach out the mentors and ask necessary doubts regarding the project implementation and their requirements.

2. **Week 1-3:**

   - Will explore the existing implementation of the application and see the points where it is failing to provide users hassle-free interface.
   - Will decide efficient Data Structures to represent the questions, trials' data.
   - Wireframing the UI and taking feedback.

3. **Week 4-6:**

   - Will discuss different algorithms for implementing choosing questions, and decide an efficient algorithm.
   - Will develop a good looking and user-friendly UI of the application.

- Will start coding the frontend in React JS with clean code practices. Will decide a CSS framework like vanilla CSS, Tailwind CSS, Material UI etc.

4. **Week 7-9:**

- Will create new Questions in easy language.
- Will implement the functions required as utilities using efficient algorithm decided before.
- Will make a basic structure, rendering questions, taking input from the user, and show Matched, Undetermined, Matched Questions.
- Authentication is taken care of

5. **Week 10-11:**

- Will implement the UI/UX of the application using the UI made on Figma.
- Will make the application responsive across various screens.
- Will understand the backend used in the existing application and integrate the APIs for fetching the JSON data of questions and trials.
- Introduce visualization features like graphs, pie charts to understand complex trial matching.

6. **Week 12-14:**

- Will integrate AI and LLM models so that user can ask relevant questions.
- AI will be used for questions generation so that users without much medical knowledge can also use the application.
- Something like chatbot can be implemented.
- Feedback from users and mentors is taken and rectify any mistakes.
- Accessibility features will be introduced.

7. **Week 15-17:**

- Will introduce PWA support and implement caching techniques.
- Will integrate the component into the existing application.
- Take feedback from mentors.

WHY ME?

I am a passionate web developer interested in various technologies, proficient in full stack. I can design in Figma, code the application based on a Figma design. I have experience in open source. Contributed to Zulip, Hacktoberfest, and various tech fests in our college. I am confident that I can be a very good part of D4CG community and contribute very well to the community if got selected.

My Tech stack:

Frontend: Html, CSS, Tailwind CSS, JavaScript, React JS, Next JS, Figma, Three JS
Backend: Node JS, Express JS, Django, Graph QL, Prisma, Mongo DB, MySQL, PostgreSQL.
App Dev: Flutter
DevOps: Git, GitHub, Linux
Languages: C, C++, JS, Python, Rust.

Hence with the proficiency and experience in frontend and backend I can contribute very well and become an asset.

My Projects:

1. Inter IIT 12.0 Trumio:

Grabbed Bronze medal in Inter IIT 12.0 hackathon in Web Development domain. Learnt a lot like team work, working for long hours.
Tech Stack used: Next JS, OAuth, OpenAI, Figma.

Website Link Github Repo

2. Personal Portfolio:

My personal portfolio, showing all the projects I have done and about me.

Tech Stack: React JS, Tailwind CSS

Website Link Code

3. **Interactive Web:**

   Beautiful animations and interactions like particle.js without using the library and from scratch html canvas.
   Tech Stack: Html Canvas

   Website Link Code

4. **Todo App:**

   A full stack todo MERN application

   Link

My availability:

I have no other commitments this summer, so GSOC will be my only priority.
I will be able to devote ample time to work on the project. I will be devoting **30-40** hours per week. So, I am applying for a **350** hours project.

Post GSOC:

Post GSOC, I would love to **contribute to D4CG** after GSOC too. I intend to be a long term contributor to D4CG.